Towards a holistic discovery of decisions in process-aware information systems

Towards a Holistic Discovery of Decisions in Process-Aware Information Systems

Johannes De Smedt*[1,2], Faruk Hasić[1], Seppe K.L.M. vanden Broucke[1], and Jan Vanthienen[1]

[1] KU Leuven Faculty of Economics and Business Department of Decision Sciences and Information Management firstname.lastname@kuleuven.be
[2] University of Edinburgh Business School Management Science and Business Economics Group johannes.desmedt@ed.ac.uk

Abstract. The interest of integrating decision analysis approaches with the automated discovery of processes from data has seen a vast surge over the past few years. Most notably the introduction of the Decision Model and Notation (DMN) standard by the Object Management Group has provided a suitable solution for filling the void of decision representation in business process modeling languages. Process discovery has already embraced DMN for so-called decision mining, however, the efforts are still limited to a control flow point of view, i.e., explaining routing (constructs) or decision points. This work, however, introduces an integrated way of capturing the decisions that are embedded in the process, which is not limited to local characteristics, but provides a decision model in the form of a decision diagram which encompasses the full process execution span. Therefore, a typology is proposed for classifying different activities that contribute to the decision dimension of the process. This enables the possibility for an in-depth analysis of every activity, deciding whether it entails a decision, and what its relation is to other activities. The findings are implemented and illustrated on the 2013 BPI Challenge log, an exemplary dataset originating from a decision-driven process.

Keywords. Decision Mining, Decision Model and Notation, Process Mining

1 Introduction

The prevalence of new works on decision modeling and mining, as witnessed by the vast amount of new works on Decision Model and Notation [1], shows an increasing interest in documenting, modeling, and analyzing the decision dimension of processes. Many research efforts have pursued the discovery of the decision layer of processes already, including the seminal work on *decision mining* [2], and its extensions and improved versions [3–6]. Nevertheless, this form of decision mining is focusing on decision point analysis, i.e., the discovery of split-operators in the control flow which are dependent on certain data variables

* Corresponding author.

tied to the activities that need to be performed subsequently. Hence, the focus still lies with the extraction of control flow information, rather than decision information. Furthermore, issues arise when dealing with loops and non-local dependencies, i.e., decision variables that are not only affecting its subsequent routing construct(s), but also other variables processed by activities further down the process. This work complements the typical decision point analysis by rather constituting the different types of activities that are present in a process model and establishing how they contribute to the decision layer of the whole model. It proposes a framework for connecting decision variables which can be linked according to any control flow representation and any data mining algorithm by constructing decision requirement diagrams. It consists of a four step approach that decides how activities are influencing variables, classifying whether they form decisions, building a decision requirement diagram, and finally, building the control flow of the process by taking into account the decisions, rather than solely the routing of activities.

This paper is structured as follows. In Section 2, an overview of decision modeling and mining is constituted to frame the problem. In Section 3, the necessities for an integrated technique are introduced and illustrated, followed by Section 4, which introduces the approach for doing so. Section 5 outlines the implementation, as well as the application of the proposed technique to the 2013 BPI Challenge log. Finally, Section 6 concludes and discusses future work.

2 Decision Modeling and Mining

This section introduces and situates the concepts used for decision modeling and mining subsequently, i.e., first decision models are elaborated and formalized, next decision mining is discussed in more detail.

2.1 Decision Models and Related Work

The decision modeling approaches present in process management literature often breach the separation of concerns between control and data flow, hence negatively influencing maintenance and reusability. They do this by hard-coding and fixing the decisions in business processes [7]. Consequently, splits and joins in business processes are misused to represent typical decision artifacts such as decision tables. Recently, the separation of processes and decision logic has become an evident trend. Such an approach is supported by the DMN standard [1], since it has the clear intention to be used in conjunction with the Business Process Model and Notation (BPMN) [8]. Decoupling decisions and processes to stimulate flexibility, maintenance, and reusability, yet integrating decision and process models is therefore of paramount importance [9]. The DMN standard allows to model and describe decisions in a declarative way on two levels, the requirements level and decision logic level. For the first level decisions requirement diagrams (DRD) are used to represent the information requirements of the decisions in the model. These diagrams can consist of several types of elements,

decisions, input data, business knowledge models, and knowledge sources. Information requirements in the DRDs represent the requirements of decisions in terms of subdecisions and input data, depicted using arrows going from the requirement to the decision. The second level uses the FEEL expression language to describe the decision logic behind every decision. The FEEL language allows to write executable decisions in a declarative language.

Besides DMN, also the Product Data Model (PDM) [10] is a well-known language to capture the dependencies that exist between decisions and their input in workflows. DMN, however, is more driven by the decision and its rationale compared to PDM, which rather focuses on the data and its impact on the workflow.

To support our approach we introduce a formal basis for decisions and requirements in DMN models. We take abstraction from the use of Business Knowledge Models and Knowledge Sources, as defined in the DMN standard. However, all definitions and theorems provided can be readily extended to include the use of these concepts.

2.1.1 Formal Definition A DMN model can be represented as follows. We adopt the definition of *decisions* and *decision requirement diagrams* from [9].

Definition 1. *A decision requirement diagram DRD is a tuple (D_{dm}, ID, IR) consisting of a finite non-empty set of decision nodes D_{dm}, a finite non-empty set of input data nodes ID, and a finite non-empty set of directed edges IR representing the information requirements such that $IR \subseteq D_{dm} \cup ID \times D_{dm}$, and ($D_{dm} \cup ID$, IR) is a directed acyclic graph (DAG).*

The DMN specification allows a DRD to be an incomplete or partial representation of the decision requirements in a decision model. The complete set of requirements is derived from the set of all DRDs in the decision model.

Definition 2. *The decision requirements level R_{DM} of a decision model DM is the set of all decisions requirement diagrams in the model.*

The information contained in this set can be combined into a single DRD representing the entire decision requirements level. The DMN standard calls such a DRD a decision requirement graph (DRG). We extend the notion of a DRG, in such a way that a DRG is a DRD which is self-contained, i.e. for every decision in the diagram all its requirements are also represented in the diagram.

Definition 3. *A decision requirement diagram $DRD \in R_{DM}$ is a decision requirement graph DRG if and only if for every decision in the diagram all its modeled requirements, present in at least one diagram in R_{DM}, are also represented in the diagram.*

According to the DMN standard a decision is the logic used to determine an output from a given input. In BPMN a decision is an activity, i.e. the act of using the decision logic. Another common meaning is that a decision is the actual result, which we call the output of a decision. We define a decision using its essential elements.

Definition 4. *A decision $d \in D_{dm}$ is a tuple (I_d, O_d, L), where $I_d \subseteq ID$ is a set of input symbols, O_d a set of output symbols and L the decision logic defining the relation between symbols in I_d and symbols in O_d.*

In case of decision tables, a commonly used reasoning construct in decision models, I_d and O_d contain the names of the input and output elements, respectively, and L is the table itself, i.e. the set of decision rules present in the table. Note that, since a DRD is a DAG, $I_d \cap O_d = \emptyset$.

2.2 Decision Mining and Related Work

In recent business process management literature, decision mining arises as a frequent term. It was first introduced in process literature in the work of [2]. The work derives and describes the routing in so-called decision points in Petri nets [11] through a decision tree algorithm. The main idea is to use the control flow data to determine the overall structure of the process first, and consequently use the instances' attributes to define where the data had an impact on the work flow. Following this seminal work, numerous other studies have been dedicated to refining decision point analysis and assessing variations of the problem [3,5,6,12]. The most holistic outcomes are provided by [13] and [4]. The former mines for read and write operations on the variables and relates them to the guards of the different activities present in a Petri net, obtaining a data-aware Petri net. The latter incorporates XOR-splits in the decision model which also consists of data attributes, which are either considered inputs or decisions themselves. This way, a combination of attributes and control flow elements is found in the form of a DMN model. A different approach is to mine for the mental actions performed by decision makers [14], captured in a Product Data Model [10].

Contrary to focusing on the control flow, other works exist that rather start from the data perspective while either incorporating control flow for clarification, or by structuring the results. In [15] a general framework for correlating business activity variables and process variables is proposed, and in [12], the resource perspective is mixed with the control flow for recommendations of future executions. In [16], Guard Stage Milestone models [17] are mined by extracting business objects and enriching them with their lifecycle information. Nevertheless, these approaches do not focus on deriving the decision rationale that is present in the process.

In [18], a framework to position all these works was proposed. This framework, depicted in Figure 1, consists of two dimensions, i.e., the decision control flow dimension and the decision model maturity dimension, to classify each approach into four quadrants. The presence of an elaborate decision control flow dimension is depicted along the vertical axis. Typical data mining approaches belong in Q1, as they do not incorporate dynamic data aspects. On the other hand, Q2 represents an approach where the primary objective is to derive the control flow of activities by fitting process models such as Petri nets [19]. Along the horizontal axis, the decision model maturity dimension is pictured. This dimension evaluates the presence of a decision model. In Q1 and Q2, no such model

is available, while a decision model is present in Q3 and Q4. Quadrants Q3 and Q4 differ in prioritisation, as in Q3 the decision model is not orthogonally connected to the process, but rather parts of the decision model are incorporated in segments of the control flow. Hence, a holistic decision model is absent in Q3. On the contrary, Q4 approaches provide a holistic decision model that incorporates all the decisions made and that can be reused throughout the process.

Clearly, the approaches building on [2] display strong abilities to extract the control flow and relating data variables to its routing elements. Other approaches provide a strong decision model output, but do not focus on how the decision was established throughout the process. Hence, there is a gap between strong control flow-driven and decision model-driven approaches, as the challenge is to develop a decision mining approach that is driven by the decision model, rather than by the control flow containing decision points. In [20] both event labels and data attributes are considered, as dependency conditions are discovered using classification and embedded in process discovery. The information on the discovered rules annote the resulting process models. Hence, this method hovers between Q3 and Q4. The approach in [16] constructs artifacts by correlating the data of the events. The control flow over these artifacts is mined as well and the outcome is presented in a holistic model containing both layers. Consequently, this is the only approach to the authors' knowledge that truly belongs in Q4, as it both handles the complexity of the data and the dynamic behavior of its activity generators.

In this paper, we will address the research gap in Q4 by introducing the ***P**rocess **M**ining **In**tegrating **D**ecisions* **(P-MInD)** framework by focusing on constructing DRDs which are compatible with process models that are activity diagram-based (e.g. Petri nets and BPMN).

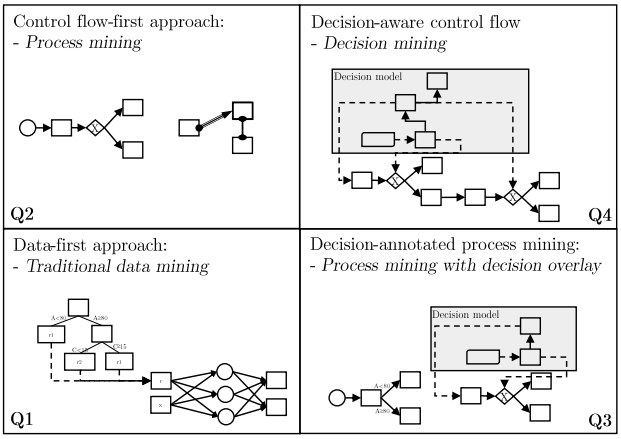


Fig. 1: The Decision Mining Quadrant.

2.2.1 Event Logs Process mining and its related techniques employ the notion of the event log to define the structure of data suitable for activity- and case-based discovery.

Definition 5. *An event log is a tuple (E, A, λ, V, var, Val, $\mathcal{L}$), where:*

- *E is a set of events.*
- *A is a set of activities (event types).*
- *$\lambda : E \to A$ is a labelling function mapping events to activities.*
- *V is a set of variables.*
- *$var : E \to 2^V$ is a function mapping events to the subset of variables used in this event.*
- *For each $v \in V$ a partial function $val_v : E \to dom_v$ mapping events to values in the domain of v. We denote the set of these partial functions as Val.*
- *$\mathcal{L} \subseteq \bigcup_{n \in \mathbb{N}} E^n$ the set of event tuples in the log.*

For brevity we use var(a) = $\{v \in var(E) | \lambda(E) = a\}$, i.e. var(a) for $a \in A$ denotes the variables for the event label with a.

Typically, special variables include the timestamp ($t \in V$) and resource ($res \in V$). The timestamp is denoted $T(e) = val_t(e)$.

Consider for example the set of events $E = \{e_1, e_2, e_3, e_4, e_5\}$, a set of variables $V = \{res, time, docid\}$, and a set of activity labels $A = \{register, send, receive\}$. Then $\mathcal{L} = \{(e_1, e_2), (e_3, e_4, e_5)\}$, with $\lambda(e_1) = register$, $\lambda(e_2) = send$, $\lambda(e_3) = receive$, $\lambda(e_4) = send$, $\lambda(e_5) = receive$, $var(e_1) = \{res, time\}$, $var(e_2, e_3) = \{res, time, docid\}$, $var(receive) = \{res, time, docid\}$, and $val_{res}(e_1) = john$, $val_{res}(e_2, e_3) = ann$, $T(e_1) = 1, T(e_2) = 4, T(e_3) = 9$.

3 Business Process Activities and Their Relation to Decisions

In this section, a typology for different activities used for making decisions in processes is proposed, as well as a running example of a decision model intertwined with a process model.

3.1 Business Activities

Decisions do not surface solely as the driver of control flow. Rather, they both encompass the routing of cases, i.e., because of decision outcomes that steer toward a certain activity tailored towards supporting its output, and the changes in the data layer of the process as well. The latter introduces numerous types of activities that are representatives of the *decision* model in the *process* model:

Definition 6. *The input and output data variables of business activities are defined as follows:*

- *$I : A \to V$, function assigning activities which receive input of a certain variable,*

– $O : A \rightarrow V$, *function assigning activities which deliver output for a certain variable.*

This enables the construction of the following activity types:

1. **Operational activities ((no) inputs, no outputs):** do not have any influence on the process' decision dimension and only act as a performer of a specific action that is tied to that specific place in the control flow. They might serve as the end of a decision. They are provided with the decision inputs needed, which are not used further in the process,
   $A_o = \{a \in A \mid O(a) = \emptyset, \}$.
2. **Administrative activities (no inputs, outputs):** have the purpose to introduce decision inputs into the process,
   $A_a = \{a \in A \mid I(a) = \emptyset \wedge O(a) \neq \emptyset\}$.
3. **Decision activities (inputs, outputs):** serve a true autonomous decision purpose as they transform decision inputs into a decision outcome,
   $A_d = \{a \in A \mid I(a) \neq \emptyset \wedge O(a) \neq \emptyset\}$.

It holds that $A_a \cup A_o \cup A_d = A$. Typically, the decision points that are used for decision mining in processes are of the decision activity type, but tailored towards deciding which activity should be performed next based on the event labels. Note that these are not included in V.

We can now make the connection with decisions and decision models.

Definition 7. *A decision in a business process can be defined as follows:*

– *A decision in a process model,* $d^a \in D_{dm}$ *is a tuple* $(I_{d_a}, O_{d_a}, L_{d_a})$*, where* $a \subseteq A_d$*,* $I_{d_a} \subseteq I(a)$*,* $O_{d_a} \subseteq O(a)$*, and* $L_{d_a} \subseteq L$*.*

This last definition connects a decision activity with a decision and it shows than one decision activity can be tied with multiple decisions. The latter implies that, within an event log, the same activity can make different decisions, i.e., changes in variable values, and can be represented as different decision nodes within a decision model, as well as different activity types. This interpretation of how activities are present in process models is the main difference with other decision mining techniques, who keep the one-to-one mapping of activities and decisions.

3.2 Running Example

Consider the example BPMN model in Figure 2 and the corresponding decision model, which illustrates the different activity types elaborated earlier, in Figure 3. The process model contains a simple control flow with an AND-split and -join, as well as an XOR-split and -join that gets repeated later on. In the first part of the model, two variables are set, i.e., *Retrieve liability (RL)* and *Retrieve category (RC)* set the liability score and risk category respectively. Since they do not contain any inputs, they are administrative activities. *Determine risk (DR)* uses those variables to set the risk score, hence it is a decision activity

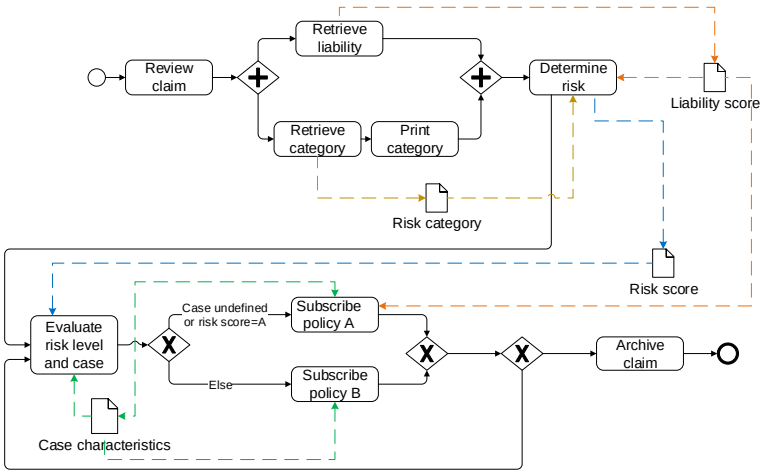


Fig. 2: BPMN model representing a liability claim process based on different decisions throughout.

with $I_D = \{liability\,score, risk\,category\}$, $O_D = \{risk\,score\}$. The risk score serves as an input for *Evaluate risk level and case (ERLC)*, which also uses the *Case characteristics (CC)* as an input (they are undefined by default). It does not have output variables, however, it decides which activity is performed next. Notice that this is the typical example of a decision point [2]. The label of the subsequent activities is not part of V, hence *ERLC* is not a decision activity, but an operational activity. *Subscribe policy B (SPB)* uses the *CC*, but has no output, hence it is an operational activity. Notice that *Subscribe policy A (SPA)* is the most convoluted activity. It serves both as a decision activity, as it sets the *CC* based on the liability score, as well as an administrative activity when it sets *CC*. Note that this is because there can exist no overlap between the inputs and outputs of decisions, hence there exist two instantiations of the activity and the model is capable of revealing how the activity contributes to the decision-making over the different iterations of the loop. Finally, *Print category (PC)* and *Archive claim (AC)* are two operational activities.

In Figure 3 the corresponding DRDs are provided. In DMN, decision activities are the typical nodes present in the model. Only a conjunction of inputs and a rationale to make decisions based on them is incorporated. Nevertheless, in this representation the administrative (indicated with $\diamond$) and operational (indicated with $\boxminus$) activities are added in gray to illustrate how the input data and all the activities in the process are related. Only *DR* and *SPA* fully process their inputs into a *Risk score (RS)*. *ERLC* has inputs, but has the label of the subsequent activities as an output. This is the typical decision point analysis approach. However, decision point analysis techniques would not be able to discover the long-distance decision dependencies of *Liability score (LS)* and *Risk category (RiC)* with their respective administrative and decision activities, as they only resolve calculations in areas of a process model that introduce XOR-gates.

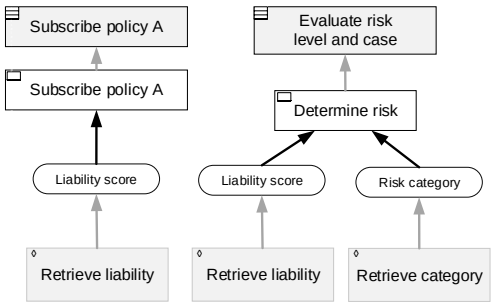


Fig. 3: The corresponding DMN model based on the process in Figure 2.

4 Discovering Decision Models

In this section, an approach is introduced to mine DRDs from event logs. The main driver of the approach is the classification of activities into the different types that were discussed previously, which are matched afterwards with how they influence the different data variables in the log.

4.1 P-MInD Approach

In order to obtain a decision model from an event log, the decisions need to be derived from the activity information in the event log first. The exact steps the approach follows are outlined in Algorithm 1.

4.1.1 Step 1: Evaluate Activity Involvement Every event is scrutinized and information regarding its variables is stored. In order to get a grasp on the effect a particular event type, i.e., activity, has on a certain variable, it is checked whether the value of a particular variable is changed. Note that the approach assumes that this data is fully and correctly recorded in the event log. This is done by comparing the previous value of the variable in the previous event $(e_{t-1} = l : T(l) < T(e), T(l) > T(f), \forall f, T(f) < T(e))$ and the current event (line 6) in case the event in question is not the start point of the particular execution trace. This is done for every event, and populates the shift metric for a certain activity. This shift is defined as $S : (V, A) \rightarrow \mathbb{N}$, i.e., the number of shifts in value of a variable $v \in V$ for an activity $a \in A$. A shift threshold, s_t, is offered to the user to adjust the sensitivity of the algorithm to take a certain variable under scrutiny. The user can also opt to exclude variables, V_u, in order to avoid the inclusion of, e.g., exogenous variables such as time stamps of events.

4.1.2 Step 2: Classify the Activities Once the influence of activities over variables is extracted, it can be used to determine whether a rationale between input and output of an activity exists. The shift metric is used to determine whether an activity actually altered the value of the variable, as outlined in Algorithm 2. If this is the case, i.e., the variable a is believed to have changed,

Algorithm 1 Mining a DMN model from an event log

```
1: procedure MINE_DMN_MOD(L, s_t, V_u, minsup)        ▷ Input: Log and parameters
2:     D_dm, ID ← ∅, S(V, A) ← 0                      ▷ Initialize variables
3:     for e ∈ L do
4:         if e_{t−1} ≠ ∅ then          ▷ Skip first event to avoid non-existing e_{t−1}
5:             for v ∈ var(e) \ V_u do              ▷ V_u excludes user-set variables
6:                 if val_v(e_{t−1}) ≠ val_v(e) then
7:                     S(v, a) + +                  ▷ Raise the variable's shift counter
8:     for a ∈ A do (D_dm, ID) ∪ buildModels(a, s_t, minsup)
9:     return buildDMNmodel(D_dm, ID)
```

then a predictive model is built which takes a as the target variable. Note that any type of predictive model can be used, e.g., decision trees, neural networks, SVMs, and so on. All other variables, i.e., $var(a) \setminus a$, are used as independent variables to determine the value of a. The downside of doing this, however, is that variables which are set together should not serve each other's predictive model as they are completely dependent of each other. This is somewhat remedied in step 3, however, cannot be fully avoided in the current approach. The evaluation of the model L is then considered to justify whether there was a causal link between the newly-set variable (v), and the other ones ($var(a) \setminus v$). For this, the Area Under Receiver Operating Curve (AUROC) value is evaluated. In case the value $L.AUROC$ is high enough, determined by the adjustable parameter $minsup$, the value is considered to be determined by the activity, which gets saved as a decision node $d^a = (v_L, v, L)$ in D_{dm} with v_L the significant independent variables of the predictive model and L the decision logic (e.g. a decision table). Note that only a singleton is considered for output, and the decisions are not multi-objective due to the nature of the predictive models used. If an activity is considered not to be a decision activity, but the event witnessed a shift nonetheless, the activity is considered to be an administrative activity, for it introduces a new value to a variable. It is stored in ID. If no shifts are made by the activity, it is considered to be an operational activity and out of scope for the decision model. Note that at any time, the corresponding decision logic is stored in L.

Algorithm 2 Constructing relations between variables of an activity

```
1: procedure BUILDMODELS(a, s_t, minsup)        ▷ Input: event log and parameters
2:     for v ∈ var(a) do
3:         if S(v, a) > s_t then
4:             L ← buildPredictiveModel(a, var(a) \ v)
5:             if L.AUROC > minsup then          ▷ Check if the model is explanatory
6:                 D_dm ← d^a = (v_L, v, L)          ▷ Save decision as decision node
7:             else ID ← (a, v)                  ▷ Save variable as input node set by a
8:     return (D_dm, ID)
```

4.1.3 Step 3: Build the Decision Model(s) Next, the elements from D_{dm} and ID need to be connected by IR to obtain a DMN model. To do so, all the inputs from ID that correspond with the inputs of the decisions in D_{dm} are connected, as well as the outputs of decisions in D_{dm} that also correspond with the inputs of other decisions. This is shown in Algorithm 3. For every relation between two decisions, it is checked whether the sequence of the relation is correct, i.e., the decision input is indeed decided before the decision is used as an input, as shown as $a_1 < a_2$ on lines 4 and 6. This can be done in numerous ways, according to, e.g., the number of times the decision delivering the input is followed by the decision using the input. This somewhat counters the effect of correlating variables that are set together, as discussed in step 2, because although they are related the check identifies whether there has been a previous value on which the shift might have been based. This can also indicate that a variable is influencing its own future value, such as is the case for CC in the running example. The number of DRDs depends on whether all components are connected, or not. Noteworthy is that multiple decisions can happen simultaneously, as an activity can set multiple variables at the same time. Control flow-based approaches do not incorporate this possibility.

Algorithm 3 Constructing the output DMN model

1: **procedure** BUILDDMNMODEL(D_{dm}, ID)
2: **for** $d_1^a = (I_1, o_1, L) \in D_{dm}$ **do**
3: **for** $d_2^a (I_1, o_1, L) \in D_{dm}$ **do**
4: **if** $o_1 \in I_2 \wedge a_1 < a_2$ **then** $IR \leftarrow (a_1, a_2)$
5: **else if** $o_2 \in I_1 \wedge a_1 > a_2$ **then** $IR \leftarrow (a_2, a_1)$
6: add all $i \in I_1$ that were not added by other decisions
7: **return** (D_{dm}, ID, IR) ▷ The DMN model as DRD

4.1.4 Step 4: Mine the Control Flow of the Decisions The final step from the P-MInD approach exists in substituting all the occurrences of the activities, once classified, with the corresponding decision nodes from the DRD in the event log. I.e., According to which values are set in a certain instantiation of the activity in the event log, the appropriate decision variant of that activity setting that value replaces the generic activity label. If multiple variables are set at the same time, they are merged in one label. This way, the control flow over the decisions can be mined directly as well. It can also be used to verify the relations between the decisions, i.e., lines 4 and 6 in Algorithm 3. Any process mining technique that mines control flow, e.g., Inductive Miner [21] can be used towards this outcome. Hence, the event log forms the source for both the decision and the process model, which gets extended with decision information.

The P-MInD approach can be considered a framework for integrated decision and process mining as numerous placeholders are present in the steps. Both the inference of connections between the different inputs and outputs of decisions, as well as the control flow perspective can be adjusted according to the most appropriate algorithms. However, the 4-step approach provides a fundamental

basis for obtaining an integrated model that contains a decision model that spans the whole control flow and in which long distance dependencies and loops in the control flow do not clutter the decision model.

4.2 Application to Running Example

Consider the running example in the case of it being recorded in an event log $\mathcal{L}$. The algorithm will first evaluate all events ($e \in \mathcal{L}$) and classify the activities. There are shifts ($S(v, a) > 0$) of variable values for activities *RL*, *RC*, *DR*, and *SPA*. In step 2, predictive models are built for all the variables for which the values shifted ($S(v, a) > s_t$). Models with a significant AUROC can be trained, i.e., D_{dm} gets $d^{DR} = (\{LS, RiC\}, RS, L_{DR \rightarrow RS})$ and $d^{SPA}(LS, CC, L_{SPA \rightarrow CC})$, in case no noise is present. The other activities with shifts are administrative activities and are considered as input nodes, i.e., ID gets $\{(RL, LS), (RC, RiC)\}$. In step 3, the models are constructed by connecting the input and output nodes of the different decision models where *RL* and *RC* serve as inputs for *DR* and *LS* serves as input for *SPA*. This way, the same DRD as in Figure 3 is obtained, without the operational activities. Finally in step 4, all the decision activities in D_{dm} replace the activity instances of *DR* and *SPA*. Note that in this case, there are no multiple variants of the activities as they only decide on one variable.

5 Implementation and Empirical Evaluation

In this section, an overview of the implementation and empirical evaluation is given. Additionally, a concise comparison with existing techniques is provided.

5.1 Implementation

The concepts of Section 4 are implemented in Java and can be used with any XES file [22]. A working prototype can be found at `http://processmining.be/PMInD`. The current models are built using the decision table plugin of the Weka data mining toolbench[3], however, any inference technique can be used to build the decision model. The output is displayed as a (set of) DRDs and can be set to display the decision logic layer in form of a decision table as well.

5.2 Evaluation

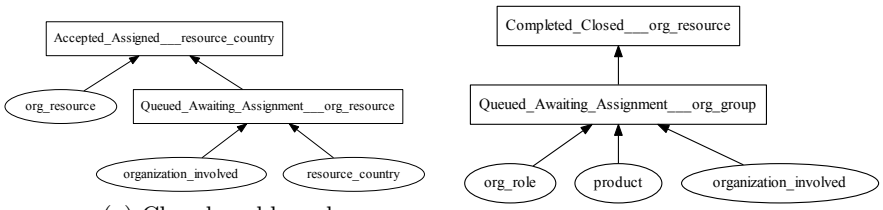
The approach was applied to the event logs made available for the 2013 BPI Challenge[4]. No extra pre-processing was performed, i.e., the unaltered files were used to create the output. The contents pertain to an incident management log at Volvo IT Belgium, in which cases are assigned a certain status according to

[3] `http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/DecisionTable.html`

[4] `http://dx.doi.org/10.4121/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07`

their nature and urgency. This type of process is typically very decision-driven and provides a suitable example to illustrate how the decision model surrounding the process can be constructed. The log is used to evaluate the first three steps of the P-MInD framework and the output can be found in Figure 4. A Petri net representing the control flow model for the closed problems log, mined with Inductive Miner (noise threshold 0.2) [21] and annotated with read and write operations (standard settings) [13] is depicted in Figure 5.

The results for the incident and closed problems logs are shown. Small, two node-DRDs are excluded from the results (hence also the DRD for the open problems variant). In the DRDs, it can be seen how the variables are actually used to decide on other variables. Furthermore, it creates a picture of how the loops should be interpreted. Many decisions in the DRDs are re-initiated many times, and it might be that, e.g., the resource's country that deals with a certain case is determined, and later redetermined, as is the case for *Queued Awaiting Assignment* and *Accepted Assigned*. Finally, multiple decisions can occur at the same time. The downside of relating dependent variables, e.g., *resource_country* and *org_resource* are frequently set together and therefore should not be included in each other's predictive model, is present. However, the DRD still explains how the re-occurrence of the decision activity *Queued_awaiting_assignment* does influence the decisions regarding the variables' values over time. This is invisible to control flow-based approaches.



(a) Closed problems log.

(b) Incident log.

Fig. 4: DRDs mined from the 2013 BPI challenge logs with $s_t = 0.1$ and $minsup = 0.8$..

Other techniques for decision (point) analysis have a hard time to capture the intrinsics of the relations between the activities as their results are still intertwined with the specific areas of a process model that contain XOR-gates. Indeed, the control flow needs many invisible activities and extra places (which constitute decision points) to tailor for the convoluted control flow, while the data, and hence the decisions, are driving the process. In Figure 5, it is illustrated that the technique of [13] has a hard time deriving interesting guards from the control flow. In Figure 4, the result of applying the approach from [4] shows that, as a result of the unclear interplay of control flow constructs, only a single DRD can be constructed because of the presence of loops and convoluted decision dependencies, as well as nominal data variables. They prevent the algorithm from finding relations between data decisions (attributes), and control flow decisions.

In this respect, P-MInD is better capable of giving insights into how the process actually evolved. By applying step 4 and replacing activities by their decision variants, no clearer control flow could be retrieved than in 5, confirming that the process is decision-driven.

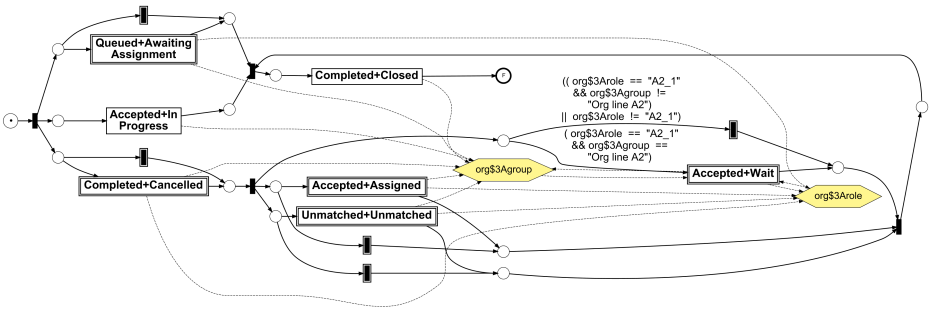


Fig. 5: Petri net mined with Inductive Miner to visualize the control flow and annotated with read and write operations.

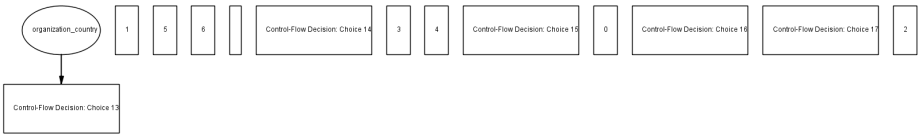


Fig. 6: DRD mined with the approach proposed in [4].

5.3 Comparison with Existing Techniques and Limitations

Contrary to [2–4], P-MInD considers the activities as the main contributors of the decision model, i.e., the decisions are made by the activities, rather than focusing on how the control flow has decided on how and where the decisions are made. Unlike P-MInD, techniques based on decision point analysis are not able to discover long-distance decision dependencies (with the exception of [4]), as they only focus on the XOR-gates of a process model. Besides, P-MInD also supports the occurrence of multiple decisions at the same time, as activities are able to set multiple variables simultaneously. Furthermore, the interplay between the *minsup* and s_t threshold provides a way to deal with noise, rather than only incorporating results of a perfectly fitting decision tree inference. The retrieval of reading and writing operations as in [13] is similar, however, the way in which they are related to the activities is different. In P-MInD, the variables are incorporated into the activities to form decisions. The focus is on the relation between the attributes through the activities, rather than towards determining the guards of the activities. While the approach of [4] is also capable of finding long distance dependencies and mining DRDs, it suffers from incorporating control flow, which can clutter up DRDs and is incapable of displaying loops. Furthermore, attributes are considered decisions, while P-MInD considers the activities as the drivers of decisions. Overall, P-MInD does not heavily rely on control flow information, although it is incorporated in steps 3 and 4 (by the $<$ relations). It adheres to the separation of concerns between decisions and processes. Hence,

P-MInD can be categorised in the fourth decision mining quadrant of Figure 1. Nevertheless, decision point analysis is compatible with P-MInD. By mining for the exact locations where certain decisions are made, the DRDs can be refined, or augmented with routing information.

The major limitation of the technique, however, stems from its independence of control flow. As illustrated before, it requires a more profound explanation of loops to avoid correlating dependent variables that are set at the same time, and does not use a strong way to incorporate sequence information in the DRD.

6 Conclusion and Future Work

This work revised the way in which a holistic decision model for process-driven environments can be retrieved. First of all, a classification of process activities was made to bridge the gap with decision model constructs. Next, an approach for retrieving DRDs based on the concept of operational, administrative, and decision activities was proposed. The approach was evaluated on the 2013 BPI Challenge log to illustrate the empirical usefulness of the framework. The results show that it is better capable of representing the decision layer of a process than existing techniques, as it does not solely rely on control flow, hence allowing for different insights into how data variables and decisions are related to activities, over long distance dependencies and loops as well.

In future endeavors, it will be investigated in what way the decision model can aid in refactoring the process model, according to the findings of [9]. This way, redesign can be suggested automatically. Furthermore, it will be tested which inference techniques are the most suitable to refine the retrieval of decision models, as P-MInD was only tested with decision table learning. Finally, while it is now assumed that the shifts hold over the whole model and can be conjoined for a global decision model, it will be investigated how an event log can be broken down according to the shifts, and the models that correspond to the particular decisions they are tied to.

References

1. OMG: Decision Model and Notation (2015)
2. Rozinat, A., van der Aalst, W.M.P.: Decision mining in prom. In: Business Process Management. Lecture Notes in Computer Science, Springer (2006) 420–425
3. Batoulis, K., Meyer, A., Bazhenova, E., Decker, G., Weske, M.: Extracting decision logic from process models. In: CAiSE. Volume 9097 of Lecture Notes in Computer Science., Springer (2015) 349–366
4. Bazhenova, E., Bülow, S., Weske, M.: Discovering decision models from event logs. In: BIS. Volume 255 of Lecture Notes in Business Information Processing., Springer (2016) 237–251
5. de Leoni, M., Dumas, M., García-Bañuelos, L.: Discovering branching conditions from business process execution logs. In: FASE. Volume 7793 of Lecture Notes in Computer Science. Springer (2013) 114–129

6. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Decision mining revisited - discovering overlapping rules. In: CAiSE. Volume 9694 of Lecture Notes in Computer Science., Springer (2016) 377–392
7. Vanthienen, J., Caron, F., De Smedt, J.: Business rules, decisions and processes: five reflections upon living apart together. In: Proceedings SIGBPS Workshop on Business Processes and Services (BPS'13). (2013) 76–81
8. OMG: Business process model and notation (BPMN) 2.0 (2011)
9. Janssens, L., Bazhenova, E., Smedt, J.D., Vanthienen, J., Denecker, M.: Consistent integration of decision (DMN) and process (BPMN) models. In: CAiSE Forum. Volume 1612 of CEUR Workshop Proceedings., CEUR-WS.org (2016) 121–128
10. Vanderfeesten, I.T.P., Reijers, H.A., van der Aalst, W.M.P.: Product based workflow support: Dynamic workflow execution. In: CAiSE. Volume 5074 of Lecture Notes in Computer Science., Springer (2008) 571–574
11. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4) (1989) 541–580
12. Kim, A., Obregon, J., Jung, J.: Constructing decision trees from process logs for performer recommendation. In: Business Process Management Workshops. Volume 171 of Lecture Notes in Business Information Processing., Springer (2013) 224–236
13. de Leoni, M., van der Aalst, W.M.: Data-aware process mining: discovering decisions in processes using alignments. In: Proceedings of the 28th annual ACM symposium on applied computing, ACM (2013) 1454–1461
14. Petrusel, R., Vanderfeesten, I., Dolean, C.C., Mican, D.: Making decision process knowledge explicit using the product data model. Lecture Notes in Business Information Processing (2011) 172–184
15. de Leoni, M., van der Aalst, W.M.P., Dees, M.: A general framework for correlating business process characteristics. In: BPM. Volume 8659 of Lecture Notes in Computer Science. Springer (2014) 250–266
16. Popova, V., Fahland, D., Dumas, M.: Artifact lifecycle discovery. International Journal of Cooperative Information Systems **24**(01) (2015) 1550001
17. Hull, R., Damaggio, E., Fournier, F., Gupta, M., Heath III, F.T., Hobson, S., Linehan, M., Maradugu, S., Nigam, A., Sukaviriya, P., et al.: Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In: Web services and formal methods. Springer (2011) 1–24
18. De Smedt, J., vanden Broucke, S.K., Obregon, J., Aekyung, K., Jung, J.Y., Vanthienen, J.: Decision mining in a broader context: an overview of the current landscape and future directions. In: Business Process Management Workshops. Lecture Notes in Business Information Processing, Springer (2016)
19. van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. Knowledge and Data Engineering, IEEE Transactions on **16**(9) (2004) 1128–1142
20. Mannhardt, F., de Leoni, M., Reijers, H., van der Aalst, W.: Data-driven process discovery: revealing conditional infrequent behavior from event logs. In: Advanced Information Systems Engineering (CAiSE), Springer (2017)
21. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from incomplete event logs. In: Petri Nets. Volume 8489 of Lecture Notes in Computer Science., Springer (2014) 91–110
22. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Xes, xesame, and prom 6. In: CAiSE Forum. Volume 72 of Lecture Notes in Business Information Processing. Springer (2010) 60–75